School Of Computer Science And Engineering

Lovely Professional University

Phagwara, Punjab

Technical Report

For AI/ML Internship Assignment

SUBJECT

Sanskrit Document Retrieval-Augmented Generation (RAG) System

SUBMITTED BY: GOVIND SINGH

REGISTRATION No: 12214199

SUBMITTED TO: IMMVERSEAI

DATE: 28 DECEMBER 2025

1. Introduction:-

This project focuses on developing a simple **Retrieval-Augmented Generation (RAG)** system using **Sanskrit documents.** The main goal of the system is to answer the user questions by first searching relevant information from the given documents and then generating a response based on that information.The Sanskrit text used in this project was provided as part of the assignment.

Retrieval-Augmented Generation works in two basic steps. First, the system searches the document collection and retrieves the most relevant text related to the user's query. After that language model uses this **retrieved text** as context to generate the answer. This approach helps the system produce more relevant responses instead of depending only on the internal knowledge of the language model.

The system was designed to run only on **CPU** as required in the assignment. No **GPU** is used in this project. Simple retrieval techniques and a small language model were selected so that the system can work properly even on limited resources. The overall implementation follows the instructions given by the company and avoids unnecessary complexity.

The purpose of this project is to gain basic understanding of how a **RAG pipeline** works in practice. It also helped in getting hands-on experience with Sanskrit text, preprocessing documents, retrieval methods, and text generation.

2. System Architecture and Flow

The system architecture of this project is based on a simple Retrieval-Augmented Generation (RAG) approach. The overall design is kept straightforward so that each part of the system can be easily understand and tested. The main focus of the system is more on functionality rather than complex optimizations.

The best part of the system is it handles document loading and preprocessing. The sanskrit documents are stored in text format and are read from the data folder. During preprocessing, unnecessary empty lines are removed and the text is divided into smaller chunks. This step make it easier to retrieve relevant information from the documents.

The next part is the retrieval module. In this step, the user's query is compared with all the text chunks generated during preprocessing. A simple vector-based retrieval method is used to find the most relevant chunks related to the query. Only top relevant chunks are selected and passed to the next stage of the system.

The final part is the generation module. The retrieved text chunks are provided as context to a lightweight language model which then generates the answer for the given question. Since the assignment requires CPU-only execution a small model is used to keep the system efficient.

Overall, the system flow of this project is summarized as:

Sanskrit Documents
↓
Preprocessing
↓
Chunking
↓
Retrieval
↓
Answer Generation

This modular structure follows standard RAG principles and helps in understanding how retrieval and generation work together in a simple pipeline.

3. Sanskrit Documents Used

The sanskrit documents used in this project was provided as part of the assignment. The document is written in Sanskrit and based on short stories. These texts were first available in documents format and then converted into plain text files using UTF-8 encoding so that Sanskrit character can be handled properly.

All the documents are stored in the data folder of the project. The content of the document is not modified and only basic cleaning is done during preprocessing. This helps in keeping the original meaning and structured of the Sanskrit text as it is.

Using text-based Sanskrit documents makes it easier to apply preprocessing, retrieval, and generation techniques. These documents finally act as the main knowledge base of the Retrieval-Augmented Generation system.

4. Preprocessing Pipeline

Before retrieval and generation the Sanskrit documents are preprossed to make them suitable for further processing. The preprocessing steps are kept simple and are implemented using basic Python code.

First, the text document is loaded from the data folder. During the cleaning step empty lines and unnecessary spaces are removed from the text. This helps in making the content more consistent and easier to handle.

After cleaning, the text is divided into smaller chunks. Chunking is done so that relevant parts of the document can be retrieved in a better way based on the user query. These chunks are later used as input for the retrieval module.

The preprocessing pipeline make sure that the original meaning of the Sanskrit text is preserved while preparing data for retrieval and generation.

5. Retrieval and Generation Mechanism

In this project, retrieval and generation are implemented as two separate steps. This makes the system easy to understand and also simple to manage.

For retrieval, the preprocessed text chunks are compared with the user question. A simple text based vector approach is used to find the most relevant chunks related to the query. Only a few top matching chunks are selected so that unnecessary information is avoided.

After retrieval, the selected text chunks are passed to the generation step. A lightweight language model is used to generate answer using the retrieved text as context. The model runs completely on CPU and does not use any GPU resources.

This approach helps in generating answers that are based on the actual content of the document rather than random text. Keeping retrieval and generation separate also follows the basic ideo of a RAG system.

6. Performance Observation and Conclusion

During testing, the system was able to retrieve relevant Sanskrit text and generate answers based on the given questions. Since the complete pipeline runs on CPU the

response time is little bit higher when compared to GPU-based system, but it is acceptable for all small datasets.

The retrieval step works well for the given documents, as the relevant chunks are correctly selected in most cases. The generated answers may not always be fully correct but they are based on the relevant content, which shows that RAG pipeline is working as expected. CPU usage remains moderate and no GPU resources are used at any stage of execution.

In conclusion, this project demonstrates a simple and working Retrieval-Augmented Generation system for Sanskrit document. The assignment requirements were followed closely and the system was implemented without unnecessary complexity. This project helped in understanding the practical working of preprocessing, retrieval and generation in a real world type scenario.